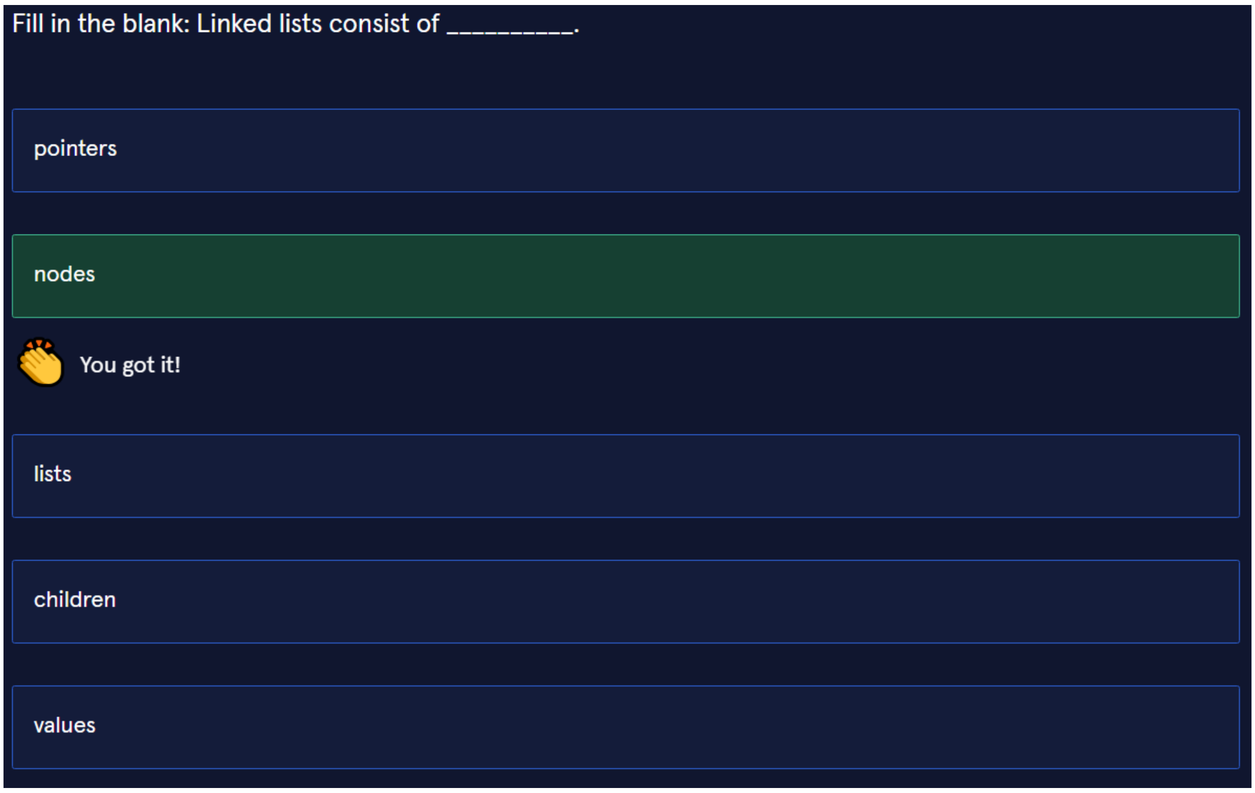
QUIZ



Fill in the blank: Linked lists consist of __________.

pointers

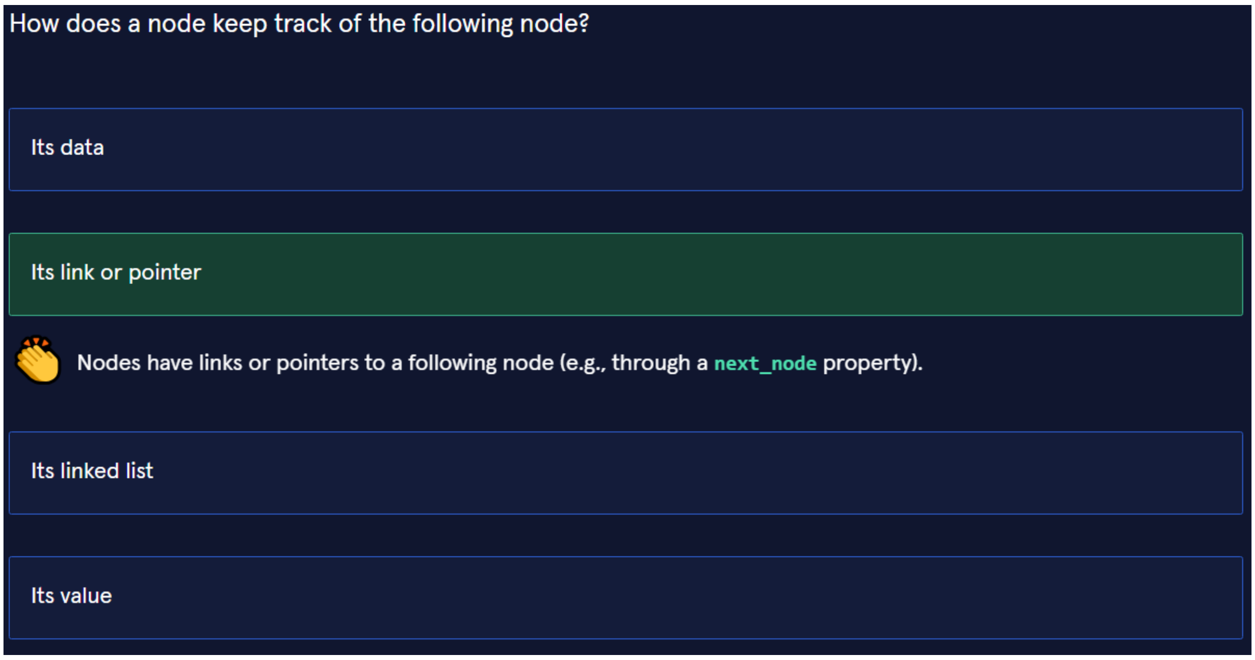
[✓ nodes]

👏 You got it!

lists

children

values



How does a node keep track of the following node?

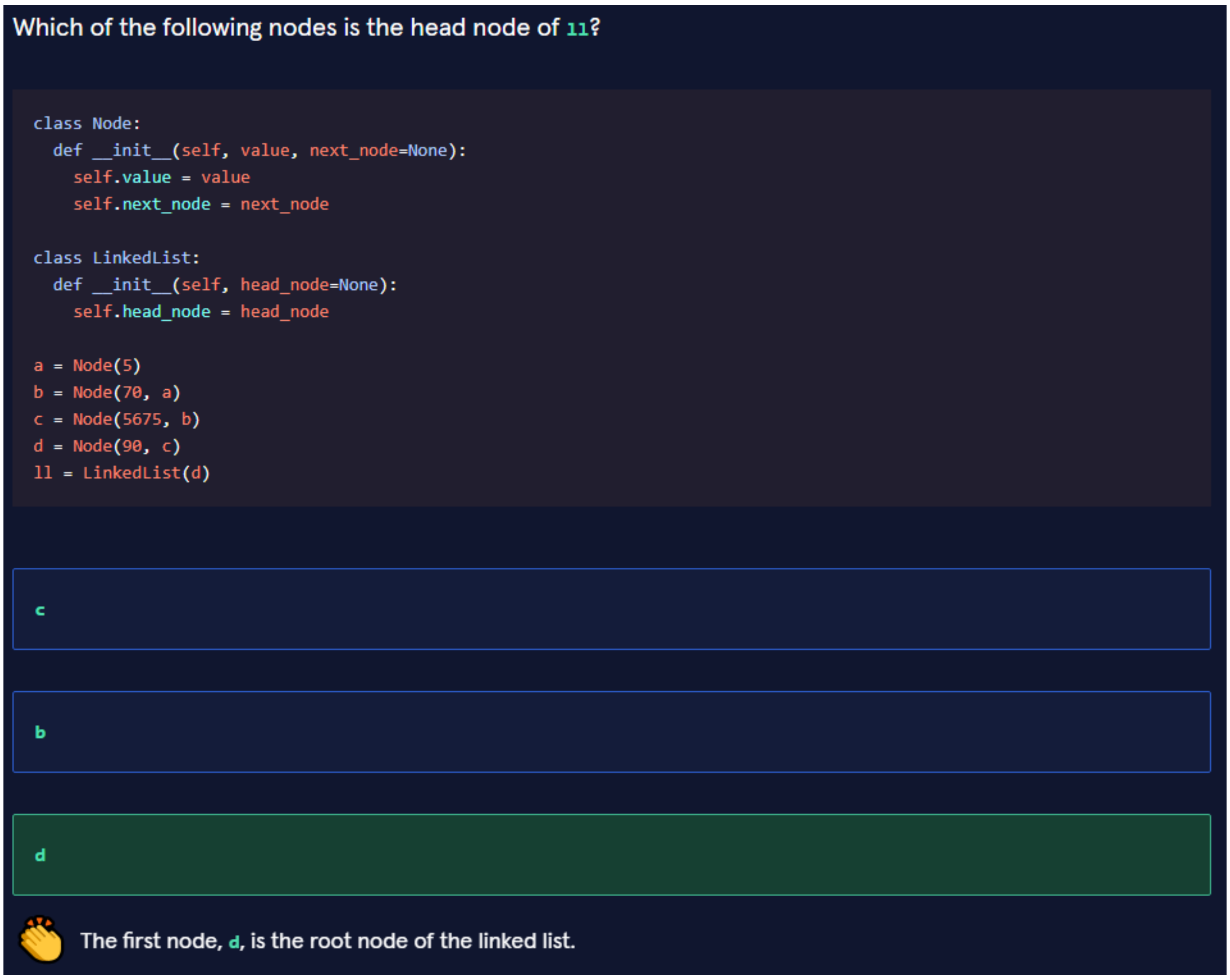
Its data

[✓ Its link or pointer]

👏 Nodes have links or pointers to a following node (e.g., through a `next_node` property).

Its linked list

Its value



Which of the following nodes is the head node of `ll`?

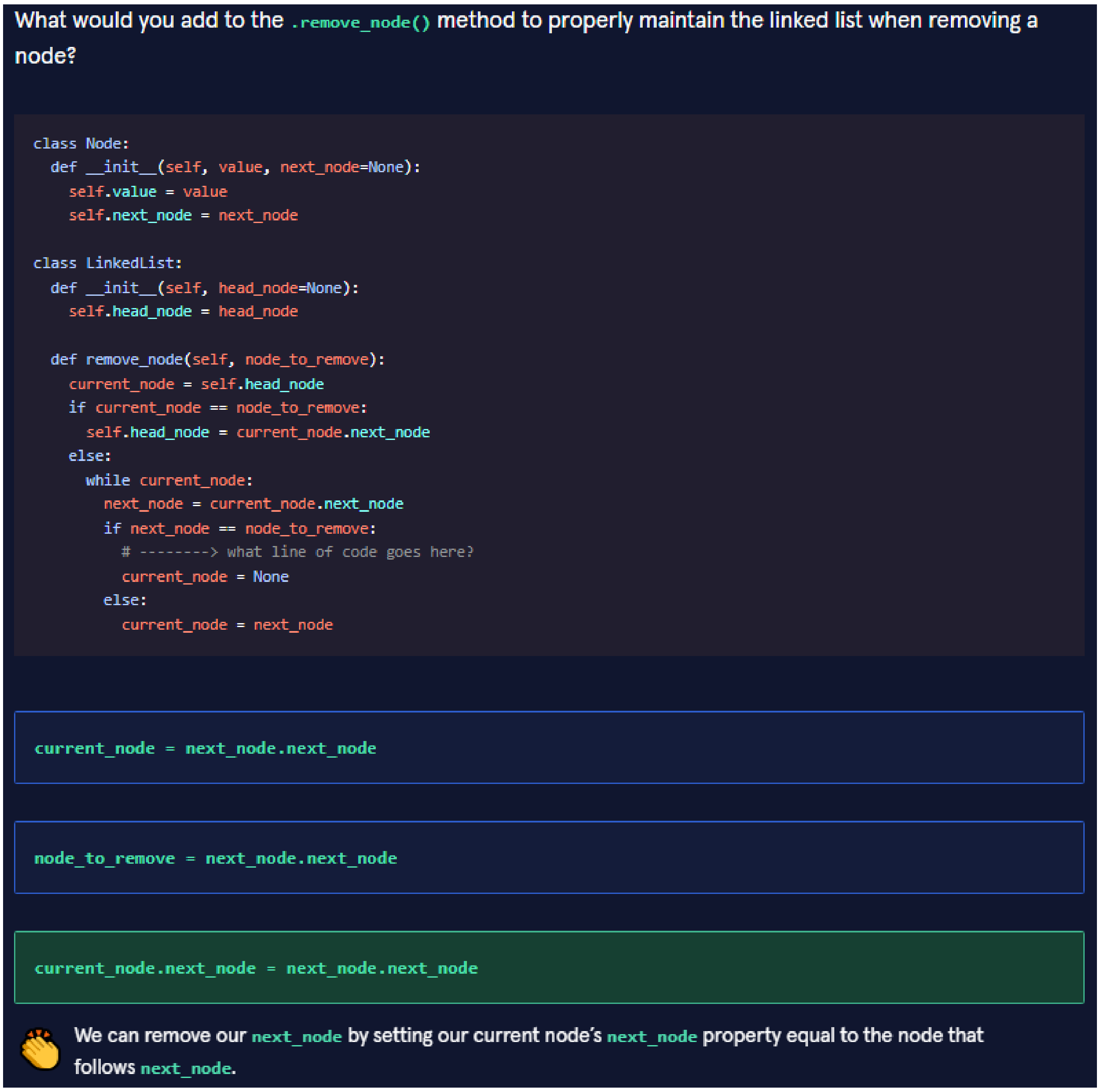
```python
class Node:
  def __init__(self, value, next_node=None):
    self.value = value
    self.next_node = next_node

class LinkedList:
  def __init__(self, head_node=None):
    self.head_node = head_node

a = Node(5)
b = Node(70, a)
c = Node(5675, b)
d = Node(90, c)
ll = LinkedList(d)
```

`c`

`b`

[✓ `d`]

👏 The first node, `d`, is the root node of the linked list.



What would you add to the `.remove_node()` method to properly maintain the linked list when removing a node?

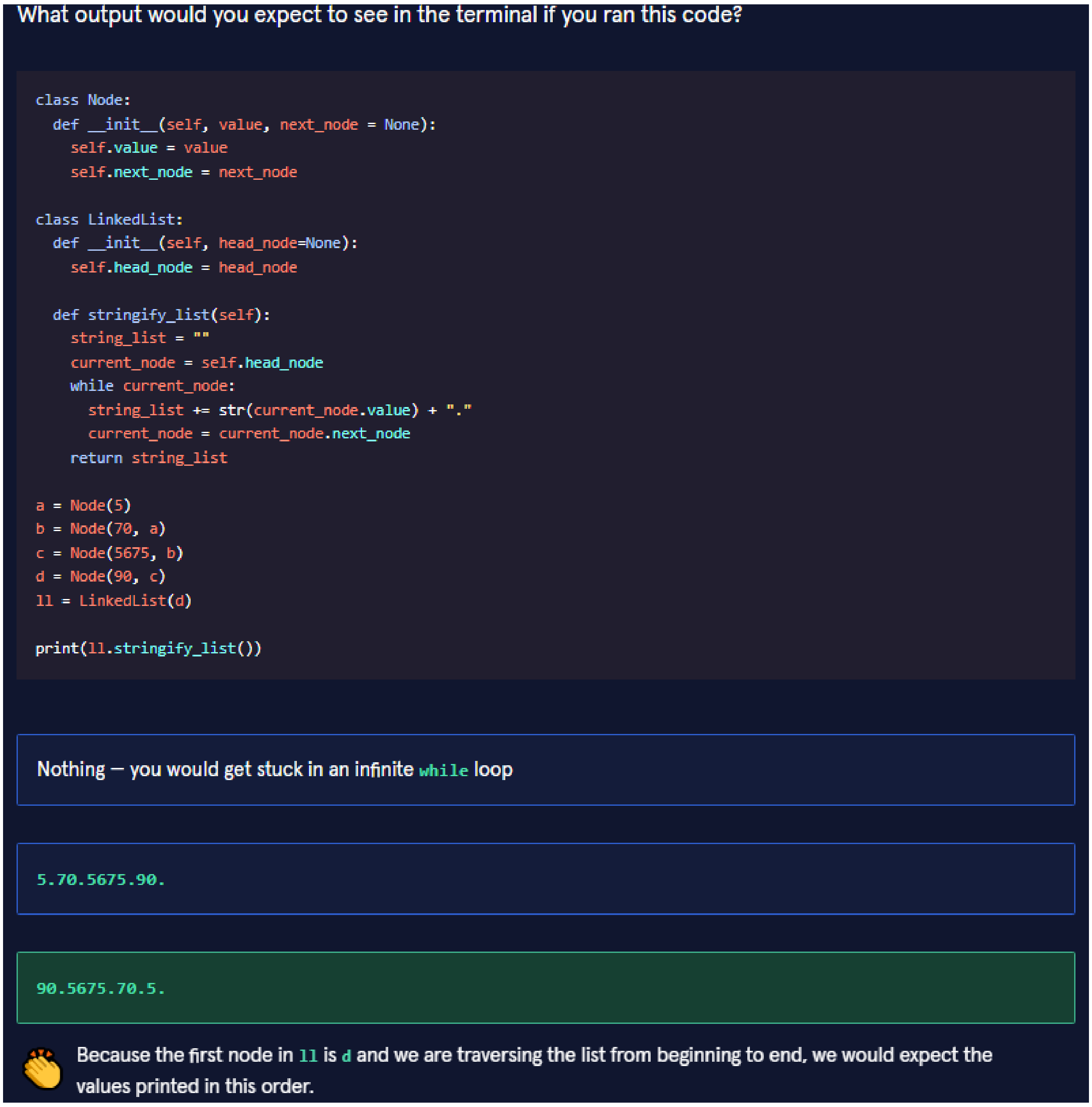
```
class Node:
  def __init__(self, value, next_node=None):
    self.value = value
    self.next_node = next_node

class LinkedList:
  def __init__(self, head_node=None):
    self.head_node = head_node

  def remove_node(self, node_to_remove):
    current_node = self.head_node
    if current_node == node_to_remove:
      self.head_node = current_node.next_node
    else:
      while current_node:
        next_node = current_node.next_node
        if next_node == node_to_remove:
          # --------> what line of code goes here?
          current_node = None
        else:
          current_node = next_node
```

`current_node = next_node.next_node`

`node_to_remove = next_node.next_node`

`current_node.next_node = next_node.next_node`

We can remove our `next_node` by setting our current node's `next_node` property equal to the node that follows `next_node`.



What output would you expect to see in the terminal if you ran this code?

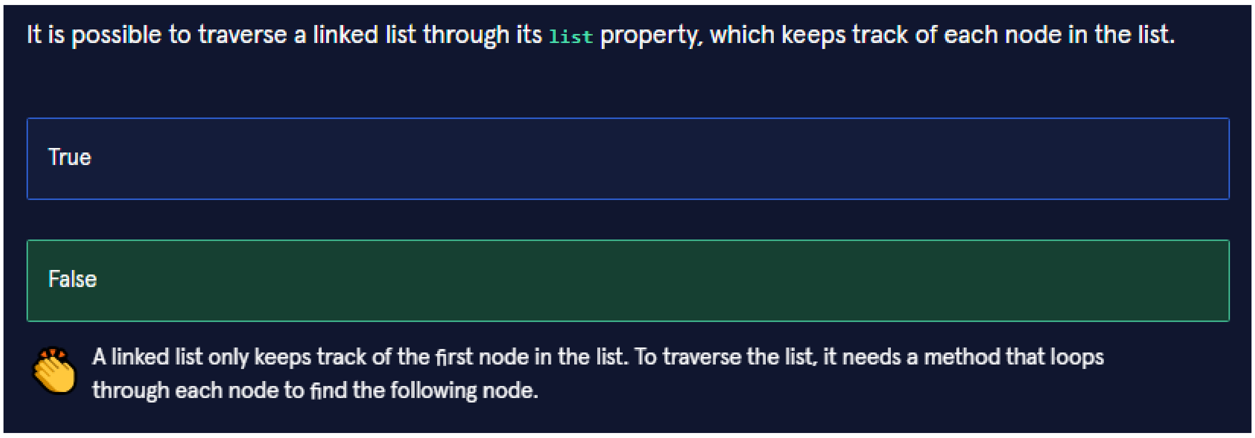
```python
class Node:
  def __init__(self, value, next_node = None):
    self.value = value
    self.next_node = next_node

class LinkedList:
  def __init__(self, head_node=None):
    self.head_node = head_node

  def stringify_list(self):
    string_list = ""
    current_node = self.head_node
    while current_node:
      string_list += str(current_node.value) + "."
      current_node = current_node.next_node
    return string_list

a = Node(5)
b = Node(70, a)
c = Node(5675, b)
d = Node(90, c)
ll = LinkedList(d)

print(ll.stringify_list())
```

Nothing — you would get stuck in an infinite `while` loop

`5.70.5675.90.`

[✓ `90.5675.70.5.`]

Because the first node in `ll` is `d` and we are traversing the list from beginning to end, we would expect the values printed in this order.

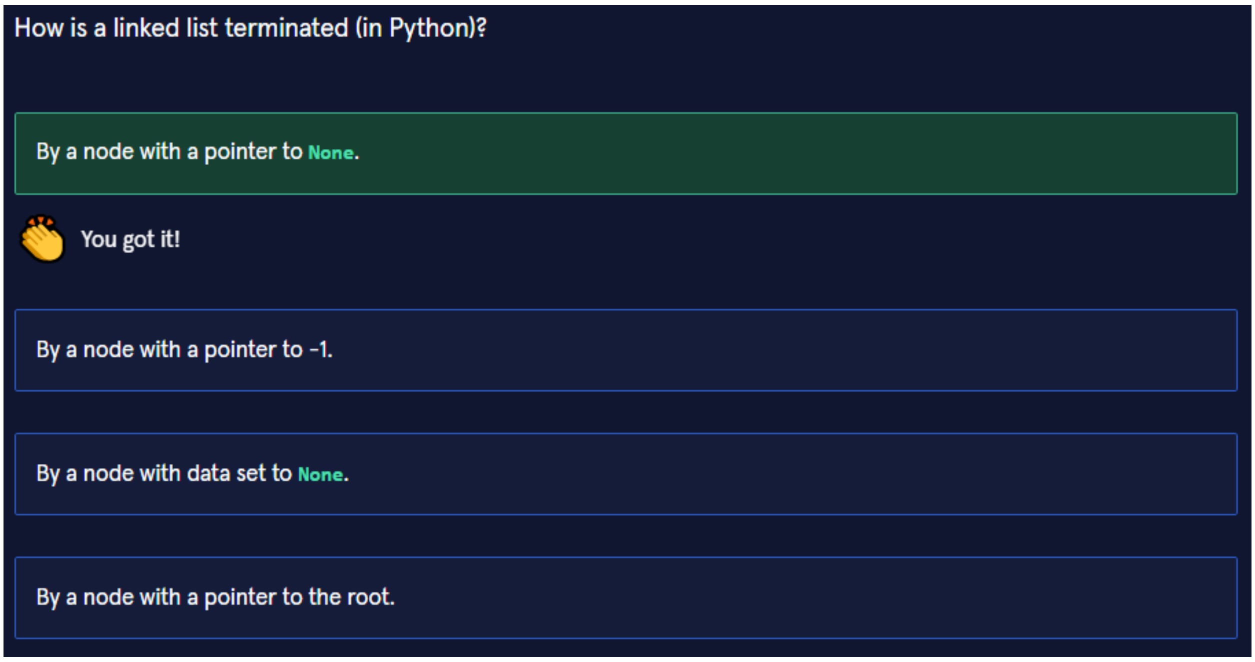


It is possible to traverse a linked list through its `list` property, which keeps track of each node in the list.

True

[✓ False]

A linked list only keeps track of the first node in the list. To traverse the list, it needs a method that loops through each node to find the following node.



How is a linked list terminated (in Python)?

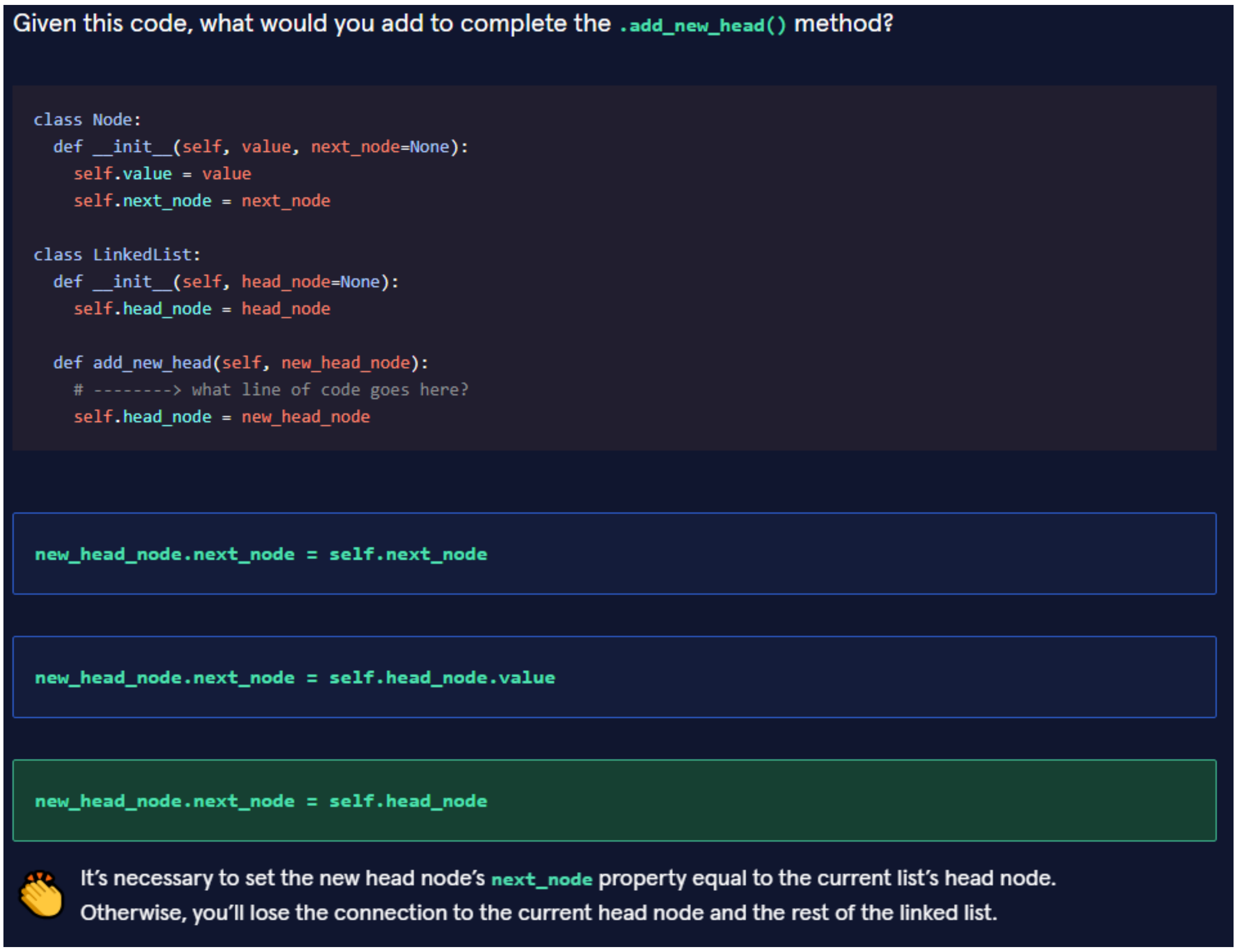
[✓ By a node with a pointer to `None`.]

You got it!

By a node with a pointer to -1.

By a node with data set to `None`.

By a node with a pointer to the root.



Given this code, what would you add to complete the `.add_new_head()` method?

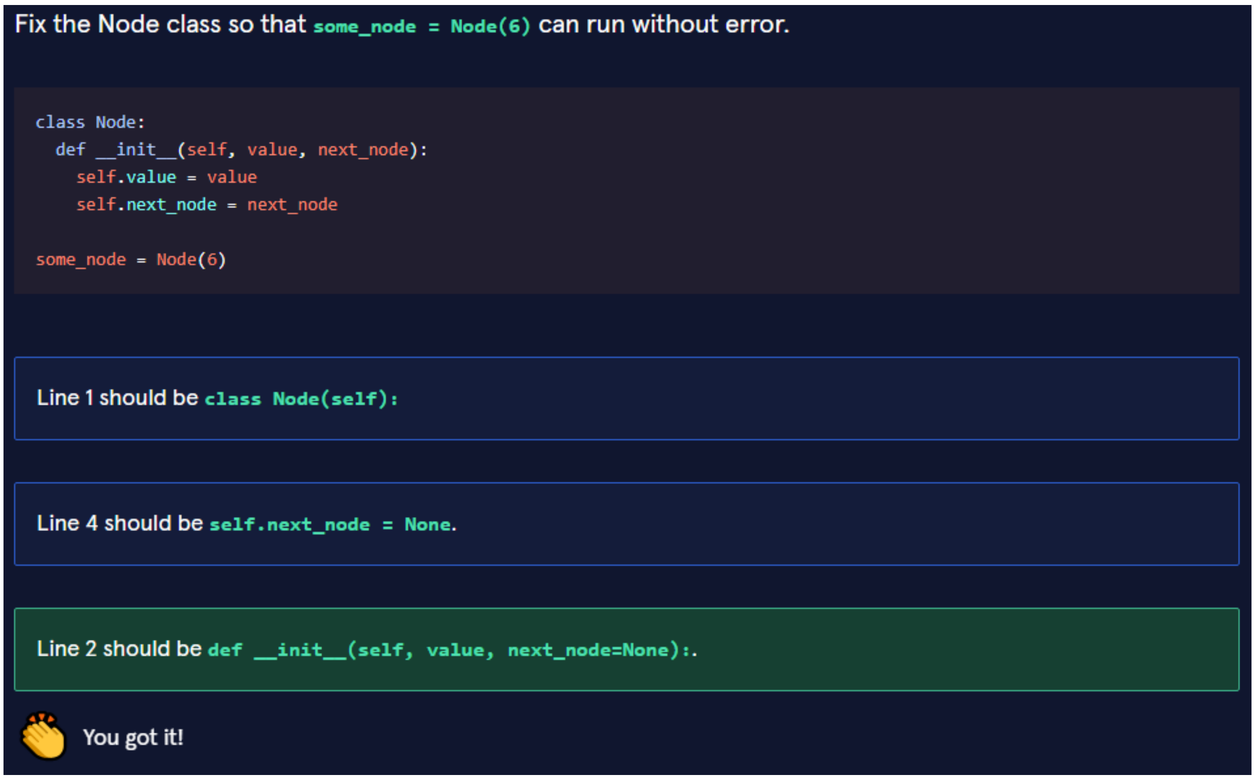
```
class Node:
  def __init__(self, value, next_node=None):
    self.value = value
    self.next_node = next_node

class LinkedList:
  def __init__(self, head_node=None):
    self.head_node = head_node

  def add_new_head(self, new_head_node):
    # --------> what line of code goes here?
    self.head_node = new_head_node
```

`new_head_node.next_node = self.next_node`

`new_head_node.next_node = self.head_node.value`

[✓ `new_head_node.next_node = self.head_node`]

It's necessary to set the new head node's `next_node` property equal to the current list's head node. Otherwise, you'll lose the connection to the current head node and the rest of the linked list.


Fix the Node class so that `some_node = Node(6)` can run without error.

```
class Node:
  def __init__(self, value, next_node):
    self.value = value
    self.next_node = next_node

some_node = Node(6)
```

Line 1 should be `class Node(self):`

Line 4 should be `self.next_node = None`.

Line 2 should be `def __init__(self, value, next_node=None):`.

👏 You got it!